A Resource-Supported Introduction to MATLAB Exercise for the Circuit Analysis Lab

L. Brent Jenkins
Kennesaw State University

Abstract

A single-period lab exercise was developed to introduce *Circuit Analysis* students to MATLAB while equipping them with the resources, knowledge, and experience they would need to complete subsequent exercises. Two supporting resources were also developed. A *MATLAB Desktop Reference* document was used to cover topics such as: creating, saving, opening, and running script files; clearing panes such as the *Command Window* or the *Workspace;* undocking and docking *MATLAB Desktop* panels; and displaying *Help*. A single-page *MATLAB Command Reference* document was created to summarize operators, functions, plot formatting options, matrix definition, row vector definition, and row vector indexing. The exercise itself was written to: encourage the use of these resources as it guides students to utilize the *Command Window* to solve basic expressions, create a script file to solve simultaneous complex equations, and prepare a well-labeled dual-trace plot to compare measured data to function-determined theoretical values.

Keywords

MATLAB circuit simultaneous complex plot

Context for the Effort

The diverse needs, resources, and constraints upon individual electrical engineering programs necessitate that mathematical analysis software be incorporated into curricula in various ways.[1-2] A distributed approach utilizing MATLAB has been adopted in our program, the starting point being a single-period, introductory exercise written for the *Circuit Analysis* lab.

Use of a single, dedicated MATLAB exercise was chosen as a means to "jump-start" student competencies without the need to significantly alter the content of other exercises. Once students have acquired basic MATLAB skills, these skills can be extended in other exercises and subsequent courses.

The absence of a programming prerequisite for *Circuit Analysis* complicated our effort to introduce MATLAB in its lab. This challenge was addressed through inclusion of additional programming-related content in the exercise.

Perhaps the novel aspect of the approach that was taken is the creation of two relatively general reference documents for students to continue to use after the completion of the introductory exercise. Thus, the exercise serves a two-fold purpose: encouraging initial use of the reference documents and guiding students through activities relevant to the course.

The *MATLAB Desktop Reference* and *MATLAB Command Reference* are provided in Appendices 2 and 3, respectively.

The *Introduction to MATLAB* Exercise

The **Prelab** identifies features in the *MATLAB Desktop* and guides the student to complete some useful customizations. It also introduces the reference documents and two means for accessing MATLAB help.

Procedure 1 introduces the use of the *Command Window* to solve basic expressions. Tasks performed include: creating a variable and assigning a value to it, using the *Workspace* to monitor the values of variables, reviewing the use of operator precedence rules, controlling the display of statements in the *Command Window*, using whitespace to improve equation readability, invoking a predefined function, using a predefined variable in the *Workspace* to access a result that was not assigned to a variable, clearing the *Command Window*, converting algebraic equations into MATLAB format, performing a parallel resistance calculation, and using the left division operator.

Procedure 2 introduces the use of a script file to solve simultaneous equations having complex coefficients. Tasks performed include: creating an m-file, abiding by the file/variable naming convention, adapting a generic matrix solution to solve a simultaneous equation, and employing whitespace to improve readability of a matrix equation in MATLAB.

Procedure 3 involves preparation of a well-labeled plot to compare measured data to function-determined theoretical values. It introduces the vector creation operator, for...next loops, array indexing, the comment operator, enumerated data storage, and two-trace linear plots. It also explores plot formatting options including line types, point types, line/point colors, descriptive titles, descriptive axis labels with units, Greek characters in axis labels, grid lines, and legends.

The *Introduction to MATLAB* exercise is provided in Appendix 1.

Results

The introductory MATLAB exercise and its accompanying reference sheets have been used by a total of 62 students—about thirty lab teams—in five sections of *Circuit Analysis* lab since Fall 2018. Given a minimal amount of instructor assistance, ninety-seven percent of these students attained a score of 73% or higher during the three-hour lab period.

References

1 MathWorks, "MATLAB Onramp," http://www.mathworks.com/learn/tutorials/matlab-onramp.html [Accessed Dec. 9, 2020]
2 Hayt Jr., W., Kemmerly, J., Phillips, J., Durbin, S., Engineering Circuit Analysis, 9th edition, McGraw-Hill, New York, NY, 2018, pp. 827-32

L. Brent Jenkins

Brent is an Associate Professor in the Electrical and Computer Engineering Department at Kennesaw State University. He has taught a wide variety of courses over the years—and has taught circuit analysis many times. One of the highlights of Brent's career has been the privilege of "being there when the lights came on" in the minds of students as they progressed through their first circuits course. He continues to seek ways to turn these lights on more efficiently and to help them to burn more brightly.

Appendix 1: ***Introduction to MATLAB*** **Exercise**

Objectives:

1. Gain familiarity with the *MATLAB Desktop*.

2. Use MATLAB operators to solve basic expressions typed into the *Command Window*.

3. Use a MATLAB script to solve a simultaneous equation having complex coefficients.

4. Create a dual-trace plot to facilitate comparison of measured and calculated data.

Equipment: Computer with MATLAB installed

Prelab: *MATLAB Desktop* Setup and Orientation

The *MATLAB Desktop* includes a *Toolstrip*, a *Quick Access* toolbar, a *Current Folder* toolbar, a *Current Folder* panel, a *Command Window* panel, and a *Workspace* panel. These features are identified pictorially by the figures in the *MATLAB Desktop Reference*.

1. Click the **Minimize Toolstrip** icon (see Table 1) on the right side of the *Toolstrip* to save display space. Right-click on the top of the *Toolstrip*—the navy-blue second-line-from-the-top—and then click the **Restore Toolstrip** button to display the entire *Toolstrip*.

2. Click the **Show Actions** icon on the *Quick Access Toolbar*, click **Customize Toolbar...**, click the plus sign to the left of *Editor-Editor* [unless it's already open], click the plus sign to the left of *New*, click-to-select the checkbox to the left of *Script*, click-to-select the checkbox to the left of **Run**, and then click OK. Thus, a **New Script** icon and a **Run** icon are added to your *Quick Access Toolbar*.

Icon Name	
Browse for Folder	📂
Help	?
Minimize Toolstrip	▲
New Script	📄
Run	▶
Show Actions	▼

Table 1: MATLAB Icons

3. Click the **Browse for Folder** icon on the *Current Folder Toolbar*. Navigate to the folder where your MATLAB script files (m-files) will be stored, and then click the **Select Folder** button to make it the *Current Folder*. Thus, the *Address Field* changes to display the complete path to the chosen location.

4. Use the *MATLAB Desktop Reference* to familiarize yourself with its basic usage.

5. Review the *MATLAB Command Reference* to acquaint yourself with the operators and functions it summarizes.

6. Click the *Command Window* and enter `help sqrt` to get basic information about the `sqrt` function. Click the *Reference page for sqrt* link to access additional information about `sqrt` in the *MATLAB Documentation* section. Click the X in the upper right corner to exit.

7. Click the **Help** icon on the *Quick Access Toolbar* to directly access *MATLAB Documentation* for more general use. The *Search Documentation* box it contains is especially useful for identifying functions for which you lack the keyword. Click X to exit.

Procedure I: Solving Expressions in the *Command Window*

1. Type `x=1.5;` into the *Command Window* and press *Enter*. Note that a variable named x having a value of 1.5 appears in the *Workspace*. Also note that a new command prompt appears in the *Command Window*.

2. Enter `y=10*x-5*x^2` into the *Command Window*. MATLAB applies standard operator precedence rules—power before multiplication before subtraction—unless parenthesis, etc. are used to override these rules. Note that the absence of a statement-terminating semicolon causes an answer to be displayed in the *Command Window*. Also note that a variable named y with a value of 3.7500 is created in the *Workspace*.

3. Enter `y = 10*x - 5*x^2`. Note that the same result produced in Step 2 is obtained despite the addition of a space to either side of the equals and minus signs. Also note that the function entered in this step is easier to read.

4. Enter `sqrt(2)`. Note that an answer is displayed in the *Command Window* while a variable named *ans* with a value of 1.4142 is created in the *Workspace*.

5. Enter `2` in the *Command Window*. Note that the value of *ans* in the *Workspace* changes to 2.

6. Enter `clc` in the *Command Window* and note that the *Command Window* is cleared.

7. Review the third sentence in Step 2, put the following lines into MATLAB command format, and enter them in the *Command Window* such that the only display other than the command prompts and instructions you typed is Req = 1.73...

$$R_1 = 2.2\times10^3$$
$$R_2 = 8.2\times10^3$$
$$R_{eq} = \left(\frac{1}{R_1} + \frac{1}{R_2}\right)^{-1}$$

8. Take **Screenshot #1** of the *Command Window* for subsequent use in your report.

9. Table 1 illustrates equation translation into MATLAB statements. Whereas theta_rad and theta_deg exemplify user-chosen variable names, *pi* is a predefined symbol for π. Enter each statement into the *Command Window* and observe the results.

Equation	MATLAB Statement
$\theta_{rad} = 2$	`theta_rad = 2;`
$\theta_{deg} = \frac{180}{\pi}\cdot\theta_{rad}$	`theta_deg = 180/pi * theta_rad`

Table 1: Translation of Equations into MATLAB Statements

10. Review the summary of the *Left Division* Operator in the *MATLAB Command Reference*. Enter `2\5` into the *Command Window* to check your understanding.

Procedure II: Solving Simultaneous Equations

1. Use the *MATLAB Desktop Reference* as needed to create an m-file in the *Current Folder*.

   File and variable names in MATLAB must start with a letter and contain only letters, numbers, and underscores. Letters are case-sensitive; embedded blanks are not permissible.

2. Adapt the program in Figure 1 to solve the following system of equations for I_2 while displaying the result in the *Command Window*.

   $$(25 - j121)\ I_1 + (-10 + j121)\ I_2 = 19.32 - j5.176$$
   $$(j121)\ I_1 + (50 - j114)\ I_2 = 0$$

   Note that z_{21}, z_{22}, and V_2 in Figure 1 are intentionally placed on the second line of their respective equations to improve readability by mimicking standard matrix and column vector formats.

```
Z = [z11 z12;
     z21 z22];
V = [V1;
     V2];
I = Z\V;
IL = I(2)
```

Figure 1: Matrix Equation Solution

3. Run your m-file and then take **Screenshot #2** of the *Editor* and *Command Window* for subsequent use in your report.

Procedure III: Creating a Dual-Trace Plot to Facilitate Comparison of Measured and Calculated Data

[The *MATLAB Command Reference* will likely prove to be an important resource during this endeavor. *MATLAB Help* should also be useful if you wish to dig deeper.]

1. Create/name an m-file and then convert the following instructions into its contents.

   Line 1:

   Use the *Vector Creation Operator* to form a vector named `RL` that starts at 0, increments by 5, and ends at 100. Consider the total number of storage locations that this instruction will produce in `RL`.

   Run the existing single-line script, double-click on `RL` in the *Workspace*, observe the contents of the `RL` vector displayed by the *Editor*, determine the total number of values stored in `RL`, close the *Variables - `RL`* half-screen, and then proceed.

   Lines 2-4:

   Establish a *for...next* loop that uses an INDEX `i`, a START value of 1, and a FINISH value based on the total number of storage locations in `RL`. Note that a start value of 0 is used by the *Vector Creation Operator* when it stores values in `RL`, while a START value of 1 is used by the *for...next* loop when it reads values from `RL`. Despite their indexing differences, both instructions need access to all locations in `RL`.

   Embed the following equation within the *for...next* loop to enable it to populate values in a `PL` vector as it steps through the values in the `RL` vector:

```
PL(i)=(100*RL(i))/((RL(i)+50)^2);   %This is Line 3
```

   Note that the m-file *Editor* will display the *comment* portion of this line in green while ignoring it for execution purposes.

Lines 5-6:

Insert the following lines "as is" to add measured data to the program:

```
RLmeas=[10, 20, 30, 40, 50, 60, 70, 80, 90, 100];

PLmeas=[0.278, 0.411, 0.475, 0.504, 0.507, 0.500, 0.486, 0.470, 0.453, 0.436];
```

The units for `RL` and `RLmeas` are Ohms; the units for `PL` and `PLmeas` are Watts.

Line 7:

Create a linear plot of `PL` versus `RL` using your choice of line type, point type, and color. Be careful to plot the correct variable on the correct axis. By implication, the variable that proceeds *versus* should be plotted on the vertical—think what it means to plot `y` versus `x`.

2. Run this seven-line script to view the resulting plot. Close the figure and proceed.

3. Add a trace that plots `PLmeas` versus `RLmeas` to the calculated value trace on the existing plot by appending—to the rightmost argument of the `plot` function—a comma and a space followed by a second set of XAXIS_VARIABLE, YAXIS_VARIABLE, and 'PLOT STRING' arguments. Select a different line type, point type, and color than those selected for display of the calculated values.

4. Run your modified script to view the dual-trace plot. Close the figure and proceed.

5. Add statements to implement each of the following objectives (in order):

   Line 8: Display a descriptive title—using words instead of symbols when applicable.

   Line 9: Display a descriptive vertical axis label (words, not symbols) followed by the appropriate paren-enclosed unit abbreviation [see instructions for Lines 5-6].

   Line 10: Display a descriptive horizontal axis label (words, not symbols) followed by the appropriate paren-enclosed unit [see instructions for Lines 5-6]. Use \Omega instead of Ohms to display **Ω** instead of Ohms.

   Line 11: Add grid lines to the plot.

   Line 12: Display a trace-identifying legend in the lower right corner.

6. Run your script and verify that it produces the desired plot.

7. Minimize the *Command Window*, dock the program-produced *Figure 1*, drag one of the overlapping central windows—either *Figure 1* or the *Editor*—to stack them in vertical halves, and then take **Screenshot #3** of these windows and save it for subsequent use in your report.

Appendix 2: *MATLAB Desktop* Reference

CLEAR

To clear the *Command Window*, right-click it and then click **Clear Command Window**.

To clear the *Workspace*, right-click it and then click **Clear Workspace**.

PANEL DOCKING/UNDOCKING

To separate a panel from the *MATLAB Desktop*, click the *Show...Actions* icon on or near the panel's top right corner, and then click **Undock**. Once undocked, a panel can be resized—even maximized—without resizing its window in the *MATLAB Desktop*.

To restore an undocked panel to the *MATLAB Desktop*, click the *Show...Actions* icon on or near the panel's top right corner, and then click **Dock**.

HELP

To display help with functions, operators, etc., type "help HELP_TOPIC" [no quotes] in the *Command Window*. For more information, type "help help"!

M-FILE CREATION AND USE

To create a new m-file if the *Editor* is closed, click the *New Script* icon on the *Quick Access Toolbar*, click **Save** on the same toolbar, select a different target folder if necessary, enter a filename, and then click **Save** on the *Select File for Save As* dialog box. *Filenames must start with a letter and contain only letters, numbers, and underscores. They are case-insensitive.*

To create a new m-file if the *Editor* is open, click on the plus-sign containing, right-most tab.

To open an existing m-file in the *Editor:* 1) Scroll down the file listing in the *Current Folder* window and double-click on the desired filename, or 2) click on the *Editor* tab or the *Home* tab, click the *Open* icon, navigate to the desired filename and then double-click.

To save an m-file that is open in the Editor: 1) click the **Save** icon in the *Quick Access Toolba*r, or 2) click the EDITOR tab on the *Toolstrip* and then click the **Save** icon. The latter option can be modified to save a copy of an existing file under a new filename by clicking the down arrow under the **Save** icon and selecting **Save As**.

To run an m-file that is open in the *Editor*: 1) Click the **Run** icon in the *Quick Access Toolba*r, or 2) Click the *Editor* tab on the *Toolstrip* and then click the **Run** icon.

To run an m-file using the prompt in the *Command Window*, type its case-sensitive filename. Only files accessible to MATLAB—i.e., those within its search path—may be invoked in this way.

To close an m-file that is open in the *Editor*, click the X to the right of its tab in the *Command Window*.

To enable an equation in an m-file to display its result(s) in the *Command Window*, omit the terminating semicolon.

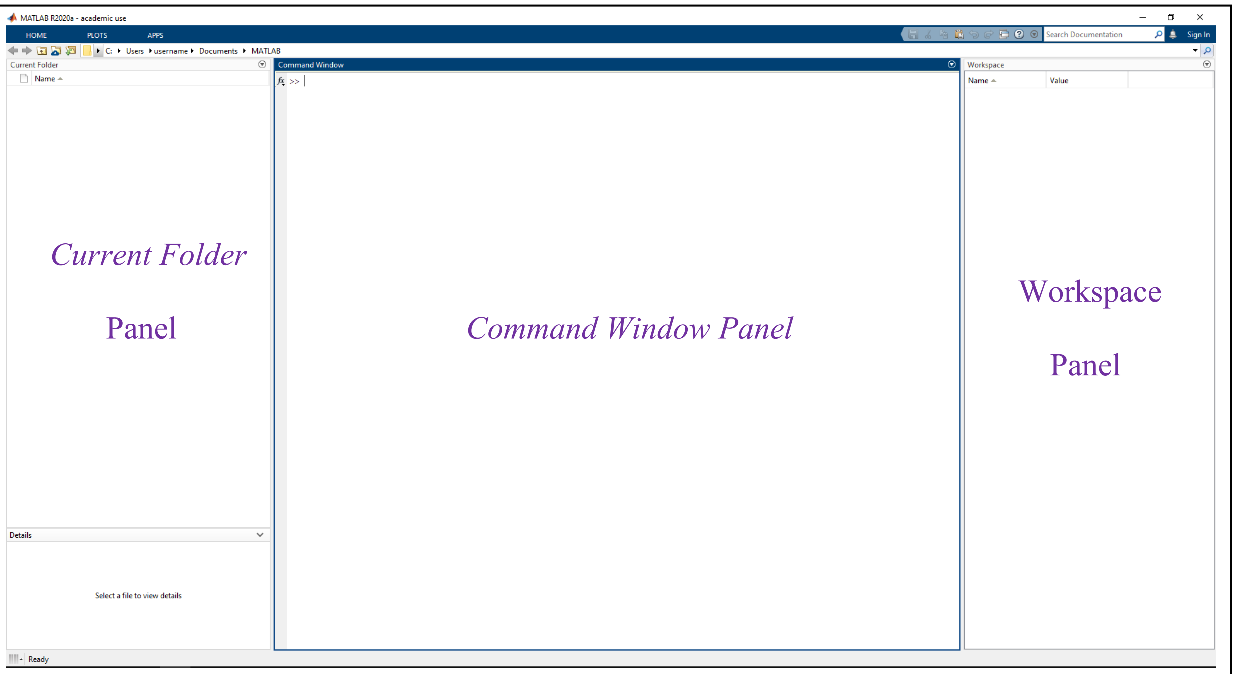

Figure 1: *MATLAB Desktop* Full-Screen

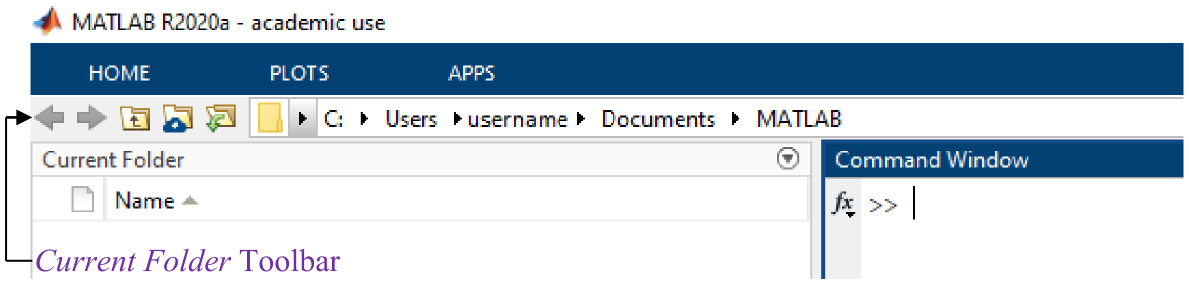

Figure 2: *MATLAB Desktop* Upper-Left Corner

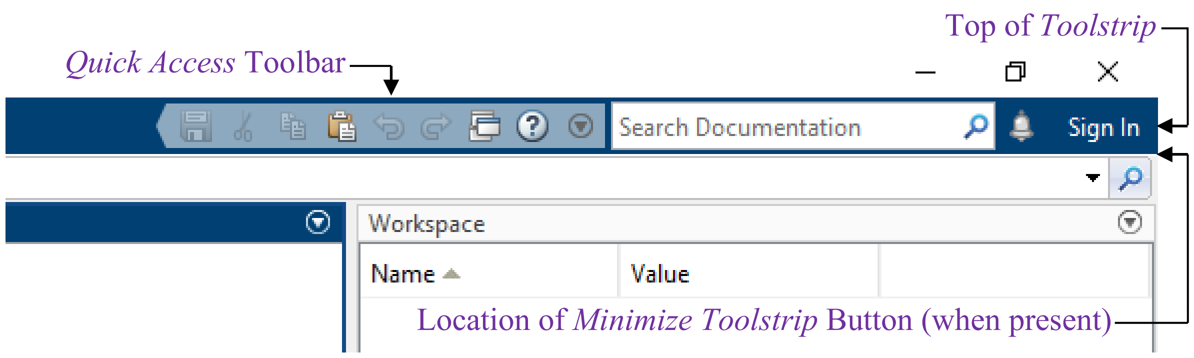

Figure 3: *MATLAB Desktop* Upper-Right Corner

Appendix 3: MATLAB Command Reference

Row Vector Definition	Indexing into a Row Vector	Matrix Definition
A = [10,20,30,40]	Z = A(3) sets Z to 30	$B = [b_{11}\ b_{12};\ b_{21}\ b_{22}]$
Equation Constant	Greek Characters in Plot Text	
pi symbolizes π	\Omega symbolizes Ω, \mu symbolizes μ, and \pi symbolizes π	

OPERATOR(S)	MATLAB
Comment [“ignore rest of line”]	%
Addition, Subtraction, Multiplication	+ - *
[Right] Division	/
Left Division	\ [e.g., Q = D\N divides N by D to find Q]
Power [e.g., $y = x^a$]	^ [e.g, y = x^a]
Exponent [e.g., 4×10^{-6}]	E [e.g., 4E-6]
Imaginary	j
Vector Creation—the Colon Operator	*v*=*s*:*i*:*e*, where vector *v* starts, increments, and ends with values *s*, *i*, and *e*, respectively
Statement Termination	;

ARITHMETIC FUNCTION(S)	MATLAB
Square Root	sqrt( )
Exponential	exp( )
Natural or Common Logarithm	log() or log10()
Trigonometric	Domain in radians: cos(), sin(), tan() Domain in degrees: cosd(), sind(), tand()
Inverse Trigonometric	Range in radians: acos(), asin(), atan( ) Range in degrees: acosd(), asind(), atand( )
Polar Components of Complex	abs() and angle()
Rectangular Components of Complex	real() and imag()

PLOTTING FUNCTION	MATLAB

Create Linear Plot
Create Plot with Semi-Log X-Axis
	plot(XAXIS_VARIABLE, YAXIS_VARIABLE, 'PLOT STRING')
	semilogx(XAXIS_VARIABLE, YAXIS_VARIABLE, 'PLOT STRING')
	where ‘PLOT_STRING’ has the form: 'LINE_TYPE POINT_TYPE COLOR' using:

LINE_TYPE		POINT_TYPE				COLOR			
Solid	-	Point	.	Downward Triangle	v	Blue	b	Magenta	m
Dotted	:	Circle	o	Upward Triangle	^	Green	g	Yellow	y
Dashdot	-.	X-mark	x	Leftward Triangle	<	Red	r	Black	k
Dashed	--	Plus	+	Rightward Triangle	>	Cyan	c	White	w
(none)		Star	*	Pentagram	p				
		Square	s	Hexagram	h				
		Diamond	d						

PLOTTING FUNCTION	MATLAB
Define Axis Scaling	axis([XMIN XMAX YMIN YMAX])
Specify Axis Limit	xlim XMAX OR ylim YMAX
Enumerate Tick Mark Locations	xticks(XTICK_LOCATIONS_ARRAY) OR yticks(YTICK_LOCATIONS_ARRAY)
Add Axis Label	xlabel('HORIZONTAL AXIS LABEL') OR ylabel('VERTICAL AXIS LABEL'
Create Plot Having Two Y Axes	yyaxis left *[create y axis on left side]* OR yyaxis right *[create y axis on right side]*
Show Grid Lines	grid on
Add Title	title('TITLE')
Add Legend	legend(‘FIRST_YAXIS_VARIABLE’,’ SECOND_YAXIS_VARIABLE’, ...) where legend position may be specified by appending ‘Location’,’COMPASS_DIR’ inside the argument and replacing ‘COMPASS_DIR’ by ‘north’,‘northeast’,’east’, etc.

ASSORTED FUNCTION	MATLAB	

For...Next Loop	for INDEX=START:FINISH
	[loop instruction(s)]
	end
		for i=1:5
		Y(i)=2*X(i)-3;
		end

Clear	clc *[Clear Command Window]*, clear *[Clear Workspace]*, clf *[Clear Current Figure]*	
Display Array	disp (ARRAY_VARIABLE_NAME)	